#Load necessary packages

library(twitteR)

library(ROAuth)

library(tidyverse)

library(purrrlyr)

library(text2vec)

library(caret)

library(glmnet)

library(ggrepel)

library(readr)

#Library for database connection

library(RODBC)

#Convert symbols (latin1) to (ASCII) with iconv

doConversion <- function(x) iconv(x, "latin1", "ASCII", "")

#Get twitter data

requestURL = "https://api.twitter.com/oauth/request\_token"

accessURL = "https://api.twitter.com/oauth/access\_token"

authURL = "https://api.twitter.com/oauth/authorize"

consumerKey = "VxxdzplmHQBOvxQHmO50BSoEQ"

consumerSecret = "ZIwaE6chaKXKkxUcmdXDNWAKviVJN3yl8Tq8971ufH3Edp1wGM"

accessToken = "77867772-lCBhVGbBoljX7KVhQTvuFb0iqkmQV52i9sW2ZWYNv"

accessSecret = "oTw9P1R2Cd57UKXmjw7d4BHIWJFnxtgUNECupKMfYpLLk"

setup\_twitter\_oauth(consumerKey, consumerSecret, accessToken, accessSecret)

#Retrieve data for Mother Teresa, store it with column names, convert the text

MotherTeresa <- twListToDF(searchTwitter('mother teresa', n = 3000, lang = 'en')) %>% dmap\_at('text', doConversion)

#Retrieve data for Donald Trump, store it with column names, convert the text

DonaldTrump <- twListToDF(searchTwitter('donald trump', n = 3000, lang = 'en')) %>% dmap\_at('text', doConversion)

#Retrieve data for Barack Obama, store it with column names, convert the text

BarackObama <- twListToDF(searchTwitter('barack obama', n = 3000, lang = 'en')) %>% dmap\_at('text', doConversion)

#Write the data into csv files, then used SSIS to move them to the database

write.csv(DonaldTrump, file = "DonaldTrump.csv", row.names = FALSE)

write.csv(BarackObama, file = "BarackObama.csv", row.names = FALSE)

write.csv(MotherTeresa, file = "MotherTeresa.csv", row.names = FALSE)

#Load tweets with column names, using read\_csv from the readr package (part of tidyverse pkg)

#Retrieved from <http://help.sentiment140.com/for-students/>

#Convert symbols, switch class values, use mutate to keep existing 0 or add 1, then return a data frame with dmap\_at. The sentiment ranges from 0 (the most negative value) to 4 (the most positive value).

readTwitterData <- read\_csv('TrainingData.csv', col\_names = c('opinion', 'userno', 'datetime', 'alias', 'diction')) %>%

#Convert symbols, switch class values, use mutate to keep existing 0 or add 1, then return a data frame with dmap\_at

dmap\_at('diction', doConversion) %>% mutate(opinion = ifelse(opinion == 0, 0, 1))

#Enable repeatable results with set.seed. Split and train data set with createDataPartition

set.seed(42)

startTraining <- createDataPartition(readTwitterData$opinion, p = 0.8, list = FALSE, times = 1)

trainTwitterFeeds <- readTwitterData[startTraining, ]

checkTwitterDatum <- readTwitterData[-startTraining, ]

#Convert to lower case with tolower. First pass process and clean up. Use tokenizer to process vector from preprocessor, then split into tokens. Iterate with itoken to create DTM. Show progress bar.

startWork <- tolower

simpleTokenizer <- word\_tokenizer

trainSet <- itoken(trainTwitterFeeds$diction, preprocessor = startWork, tokenizer = simpleTokenizer, id = trainTwitterFeeds$userno, progressbar = TRUE)

checkSet <- itoken(checkTwitterDatum$diction, preprocessor = startWork, tokenizer = simpleTokenizer, id = checkTwitterDatum$userno, progressbar = TRUE)

#Build dictionary of terms with create\_vocabulary. Define with vocab\_vectorizer how to map the word to indices. Create\_dtm creates the DTM (Document Term Matrix)

dictionary <- create\_vocabulary(trainSet)

dictionaryVector <- vocab\_vectorizer(dictionary)

docTermMatrixTraining <- create\_dtm(trainSet, dictionaryVector)

docTermMatrixChecking <- create\_dtm(checkSet, dictionaryVector)

#tf-idf modelling by term-frequency and inverse-document-frequency, to be used to align the model

modelTfidf <- TfIdf$new()

#Align the data model to the set that was trained with fit\_transform. Change the trained data with the model.

docTermMatrixTraining\_modelTfidf<- fit\_transform(docTermMatrixTraining, modelTfidf)

docTermMatrixChecking\_modelTfidf<- fit\_transform(docTermMatrixChecking, modelTfidf)

#Train the model and use k-fold cross validation with cv.glmnet. Check under the ROC curve. Use nfolds = 5 for faster training. Set maxit = 1e3 to keep iterations low for faster training.

modelTrainingWithGlmnet <- cv.glmnet(x = docTermMatrixTraining\_modelTfidf, y = trainTwitterFeeds[['opinion']], family = 'binomial', alpha = 1, type.measure = "auc", nfolds = 5, thresh = 1e-3, maxit = 1e3)

#Plot model and make prediction. Prediction is made with the trained model using the defined object type. Create and test for numeric object before auc

plot(modelTrainingWithGlmnet)

makePrediction <- predict(modelTrainingWithGlmnet, docTermMatrixChecking\_modelTfidf, type = 'response')[ ,1]

auc(as.numeric(checkTwitterDatum$opinion), makePrediction)

#Reduce the time to rerun the model by saving it

saveRDS(modelTrainingWithGlmnet, 'modelTrainingWithGlmnet.RDS')

########################################################################################################################################################################################

#Create database connection. Get data from the CheckText database in the DataToAnalyze table.

con <- odbcDriverConnect(connection="Driver={SQL Server Native Client 11.0};server=localhost;database=CheckText;trusted\_connection=yes;")

DataToAnalyzeFrDb <- sqlFetch(con,"DataToAnalyzeDonaldTrump",colnames = FALSE)

DataFromDatabase <- DataToAnalyzeFrDb %>% dmap\_at('diction', doConversion)

########################################################################################################################################################################################

#Tokenize and start processing

tweetDiction <- itoken(DataFromDatabase$diction, preprocessor = startWork, tokenizer = simpleTokenizer, id = DataFromDatabase$userno, progressbar = TRUE)

tweetsDocTermMatrix <- create\_dtm(tweetDiction, dictionaryVector)

#use tf-idf to transform the data

tweetsDocTermMatrix\_modelTfidf<- fit\_transform(tweetsDocTermMatrix, modelTfidf)

#The model being classified is loaded below. Try to predict what is the probability that the statements will be positive. Append the column with the rates to the data from the database

modelTrainingWithGlmnet <- readRDS('modelTrainingWithGlmnet.RDS')

probabilityOfPositivity <- predict(modelTrainingWithGlmnet, tweetsDocTermMatrix\_modelTfidf, type = 'response')[ ,1]

DataFromDatabase$opinion <- probabilityOfPositivity

########################################################################################################################################################################################

#To prevent warning messages, which states that some objects are masked from the dataset.

detach(DataFromDatabase)

#Create graph with labels at the distribution points

attach(DataFromDatabase)

#Create changing colors

changingColors <- colorRampPalette(c("red", "yellow", "blue"))

#Add colors as column to dataset per opinion values

DataFromDatabase$ColorColumn <- changingColors(10)[as.numeric(cut(DataFromDatabase$opinion, breaks = 10))]

#Run plot with changing color points and use stars

plot(ResponseDate, opinion, main=" Probability of reading positive words about DonaldTrump on Twitter.", xlab="Response Dates", ylab="Opinion Scores", pch = 8, col = DataFromDatabase$ColorColumn)

abline(h =0.5, col="green")

#Put legend to the top right

legend("topright", legend = c("Negative", "Neutral", "Positive"), col = DataFromDatabase$ColorColumn, pch = 8, bty = "o", pt.cex = 1, cex = 1, text.col = DataFromDatabase$ColorColumn, horiz = F, inset = c(0.0001, 0.0001))

########################################################################################################################################################################################

########################################################################################################################################################################################

#Create database connection. Get data from the CheckText database in the DataToAnalyze table.

con <- odbcDriverConnect(connection="Driver={SQL Server Native Client 11.0};server=localhost;database=CheckText;trusted\_connection=yes;")

DataToAnalyzeFrDb <- sqlFetch(con,"DataToAnalyzeMotherTeresa",colnames = FALSE)

DataFromDatabase <- DataToAnalyzeFrDb %>% dmap\_at('diction', doConversion)

########################################################################################################################################################################################

#Tokenize and start processing

tweetDiction <- itoken(DataFromDatabase$diction, preprocessor = startWork, tokenizer = simpleTokenizer, id = DataFromDatabase$userno, progressbar = TRUE)

tweetsDocTermMatrix <- create\_dtm(tweetDiction, dictionaryVector)

#use tf-idf to transform the data

tweetsDocTermMatrix\_modelTfidf<- fit\_transform(tweetsDocTermMatrix, modelTfidf)

#The model being classified is loaded below. Try to predict what is the probability that the statements will be positive. Append the column with the rates to the data from the database

modelTrainingWithGlmnet <- readRDS('modelTrainingWithGlmnet.RDS')

probabilityOfPositivity <- predict(modelTrainingWithGlmnet, tweetsDocTermMatrix\_modelTfidf, type = 'response')[ ,1]

DataFromDatabase$opinion <- probabilityOfPositivity

########################################################################################################################################################################################

#To prevent warning messages, which states that some objects are masked from the dataset.

detach(DataFromDatabase)

#Create graph with labels at the distribution points

attach(DataFromDatabase)

#Create changing colors

changingColors <- colorRampPalette(c("red", "yellow", "blue"))

#Add colors as column to dataset per opinion values

DataFromDatabase$ColorColumn <- changingColors(10)[as.numeric(cut(DataFromDatabase$opinion, breaks = 10))]

#Run plot with changing color points and use stars

plot(ResponseDate, opinion, main=" Probability of reading positive words about Mother Teresa on Twitter.", xlab="Response Dates", ylab="Opinion Scores", pch = 8, col = DataFromDatabase$ColorColumn)

abline(h =0.5, col="green")

#Put legend to the top right

legend("topright", legend = c("Negative", "Neutral", "Positive"), col = DataFromDatabase$ColorColumn, pch = 8, bty = "o", pt.cex = 1, cex = 1, text.col = DataFromDatabase$ColorColumn, horiz = F, inset = c(0.0001, 0.0001))

########################################################################################################################################################################################

########################################################################################################################################################################################

#Create database connection. Get data from the CheckText database in the DataToAnalyze table.

con <- odbcDriverConnect(connection="Driver={SQL Server Native Client 11.0};server=localhost;database=CheckText;trusted\_connection=yes;")

DataToAnalyzeFrDb <- sqlFetch(con,"DataToAnalyzeBarackObama",colnames = FALSE)

DataFromDatabase <- DataToAnalyzeFrDb %>% dmap\_at('diction', doConversion)

########################################################################################################################################################################################

#Tokenize and start processing

tweetDiction <- itoken(DataFromDatabase$diction, preprocessor = startWork, tokenizer = simpleTokenizer, id = DataFromDatabase$userno, progressbar = TRUE)

tweetsDocTermMatrix <- create\_dtm(tweetDiction, dictionaryVector)

#use tf-idf to transform the data

tweetsDocTermMatrix\_modelTfidf<- fit\_transform(tweetsDocTermMatrix, modelTfidf)

#The model being classified is loaded below. Try to predict what is the probability that the statements will be positive. Append the column with the rates to the data from the database

modelTrainingWithGlmnet <- readRDS('modelTrainingWithGlmnet.RDS')

probabilityOfPositivity <- predict(modelTrainingWithGlmnet, tweetsDocTermMatrix\_modelTfidf, type = 'response')[ ,1]

DataFromDatabase$opinion <- probabilityOfPositivity

########################################################################################################################################################################################

#To prevent warning messages, which states that some objects are masked from the dataset.

detach(DataFromDatabase)

#Create graph with labels at the distribution points

attach(DataFromDatabase)

#Create changing colors

changingColors <- colorRampPalette(c("red", "yellow", "blue"))

#Add colors as column to dataset per opinion values

DataFromDatabase$ColorColumn <- changingColors(10)[as.numeric(cut(DataFromDatabase$opinion, breaks = 10))]

#Run plot with changing color points and use stars

plot(ResponseDate, opinion, main="Probability of reading positive words about Barack Obama on Twitter.", xlab="Response Dates", ylab="Opinion Scores", pch = 8, col = DataFromDatabase$ColorColumn)

abline(h =0.5, col="green")

#Put legend to the top right

legend("topright", legend = c("Negative", "Neutral", "Positive"), col = DataFromDatabase$ColorColumn, pch = 8, bty = "o", pt.cex = 1, cex = 1, text.col = DataFromDatabase$ColorColumn, horiz = F, inset = c(0.0001, 0.0001))

########################################################################################################################################################################################